Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

DHT-based lightweight broadcast algorithms in large-scale computing infrastructures

Kun Huang [a], Dafang Zhang [b,*]

[a] School of Computer and Communication, Hunan University, Changsha, Hunan Province 410082, PR China

[b] School of Software, Hunan University, Changsha, Hunan Province 410082, PR China

ARTICLE INFO

Article history:
Received 20 June 2008
Received in revised form
12 August 2009
Accepted 29 August 2009
Available online 6 September 2009

Keywords:
Grid
Peer-to-Peer network
Distributed hash table
Broadcast
Load balancing

ABSTRACT

Scalable and efficient broadcast is essential to large-scale computing infrastructures such as PlanetLab and Grids. Previous broadcast algorithms exploit the greedy routing mechanisms of a Distributed Hash Table (DHT) to achieve the scalability. However, they suffer from load unbalancing and high overhead for constructing and maintaining a distributed broadcast tree (DBT). This paper presents DHT-based lightweight broadcast algorithms to overcome these limitations. Our algorithms leverage the topology and routing mechanism of Chord to select appropriate children of each node in a top-down approach. According to the node identifier distribution of Chord, we propose two broadcast algorithms over DHT. When nodes are uniformly distributed in the identifier space, a token-based broadcast algorithm is proposed, where each node selects the finger nodes as its children by a token value. When nodes are arbitrarily distributed in the identifier space, a partition-based broadcast algorithm is proposed, where each node partitions its identifier space into two subspaces and selects the agent nodes in the subspaces as its children. We show theoretically and experimentally that both token-based and partition-based algorithms can implicitly build a balanced DBT from the novel routing paths of DHT, where the branching factor is at most two and the tree height is $O(\log n)$ in a Chord of n nodes, without extra space for storing children and additional overhead for explicitly maintaining the parent–child membership.

1. Introduction

Large-scale computing infrastructures such as PlanetLab [1], E-science Grids [2], and Enterprise Desktop Grids [3] have become increasingly important for developing and deploying many emerging distributed applications such as content distribution networks, peer-to-peer systems, distributed games, and scientific computing. These computing infrastructures typically consist of large numbers of personal workstations and dedicated servers scattered around the world. For example, PlanetLab (2008) currently consists of 870 nodes at 460 sites [4], and the planet-scale Grid has 100,000 CPUs, mostly PCs and workstations [5]. As the size of the computing infrastructures continues to grow, it is very challenging for administrators to efficiently manage such large-scale dynamic distributed systems.

Distributed information management systems [6–9] have been extensively used in the large-scale computing infrastructures for a broad range of network services such as network monitor and management, service placement, resource management and task scheduling, and content distribution. Distributed broadcast is one of the fundamental primitives of distributed information management systems to disseminate information on a global scale. For instance, researchers replicate their programs on tens of thousands of nodes before launching a distributed application, and administrators distribute software releases and patch updates on all the federated nodes. Thus it is required to perform scalable and efficient content dissemination across large-scale computing infrastructures.

In recent years, Peer-to-Peer (P2P) based broadcast algorithms have been proposed to achieve scalability. There are two design principles for a P2P-based broadcast algorithm according to overlay structures: tree-based and mesh-based approaches. The tree-based approach constructs a tree overlay as a content delivering structure, such as ESM [10], NICE [11], Scribe [12], and Bayeux [13]. Since the single tree structure is vulnerable to failure of an interior node, multiple-tree structures such as SplitStream [14] and CoopNet [15] are proposed to improve the resilience, where each substream of content is delivered along one of multiple disjoint trees. On the other hand, the mesh-based approach, such as Bullet [16], FastReplica [17], Bullet' [18], BitTorrent [19], and CoBlitz [20], constructs a data-driven mesh overlay as a swarm system, where each node has a small set of neighbors to exchange data. Despite

* Corresponding author. Tel.: +86 0731 88821570.
E-mail addresses: huangkun@hunu.edu.cn (K. Huang), dfzhang@hunu.edu.cn (D. Zhang).

these arguments [21,22] against the two approaches, the tree-based approach is more suitable for large-scale computing infrastructures. This is because there is a large fraction of relatively stable dedicated nodes, and the tree structure has the simplicity and controlled overhead. Hence, we mainly focus on the tree-based approach to broadcasting in large-scale computing infrastructures.

Structured P2P systems have been recently proposed, such as Chord [23], CAN [24], Pastry [25], and Tapestry [26]. These structured systems use a Distributed Hash Table (DHT) as the routing substructure, with good characteristics of scalability, load balancing, and fault tolerance. Thus, DHT-based broadcast is a promising way in large-scale computing infrastructures. It is critical and crucial for a DHT-based broadcast algorithm to meet scalability, robustness, and load balancing. First, the algorithm should scale well to a large number of participating nodes with only a limited number of messages passing. Also it should have a low overhead of constructing and maintaining a distributed broadcast tree (DBT). Second, the algorithm should be robust to dynamics of node arrivals and departures. Finally, the algorithm should ensure good load balancing in the sense that the broadcast workload is evenly distributed among all participating nodes in the DBT, without any bottlenecks or hotspots of content delivery. Hence, load balancing is essential for scalability and robustness of a P2P-based broadcast algorithm.

Most DHT-based broadcast algorithms [6–8,12,14,15,27–32] have been recently proposed to support efficient broadcasting on large scale. These broadcast algorithms use the routing mechanisms of DHT to build up an overlay routing path between a source node and each destination node. Then a DBT rooted at the source node is constructed by merging these routing paths. A DHT-based broadcast algorithm has two approaches to constructing a DBT: top-down and bottom-up approaches. For example, the k-ary search based broadcast algorithm [31] adopts the top-down approach, where starting with a source node, each node selects its appropriate children, while the reverse-path forwarding based broadcast algorithm [12] adopts the bottom-up approach, where starting with all destination nodes, each node selects its appropriate parent to a source node. The bottom-up approach facilitates both aggregation and broadcast in the same DBT but consumes extra overhead on a node churn, while the top-down approach has low overhead due to not explicitly maintaining the child-parent membership. In this paper, our algorithms build a balanced DBT in a top-down approach.

However, existing DHT-based broadcast algorithms suffer from load unbalancing and high overhead. First, DHT is a greedy routing algorithm. In the DHT, each node always forwards a searched key to the closest preceding node in its finger table, whose identifier is closer to the key in the identifier space. The greedy essence of DHT results in that previous DHT-based broadcast algorithms, such as the k-ary search based broadcast algorithm [31] and the reverse-path forwarding based broadcast algorithm [12], construct a flat and unbalanced DBT. Such a DBT causes a node to be a performance bottleneck and a single failure on broadcasting. Second, these DHT-based broadcast algorithms have high overhead of constructing and maintaining a DBT with respect to a large number of participating nodes. For example, the reverse-path forwarding based broadcast algorithm [12] requires interior nodes to consume extra space for storing their children on reverse forwarding, which leads to additional maintenance overhead on node dynamics. Furthermore, these algorithms typically adopt the pushdown and anycast methods [11,13] to tackle the node overloading by adjusting the branching factors between nodes in a DBT. But, Bharambe et al. [41] indicated that these adjustment methods result in a significant number of non-DHT links that are present in the DBT but are not part of the routing links of DHT. The non-DHT links not only restrict the scalability of DBT, but also incur higher maintenance overhead in the DBT due to the dynamic nodes.

To address above issues, this paper proposes DHT-based lightweight broadcast algorithms in the large-scale computing infrastructures to achieve both scalability and load balancing. Our broadcast algorithms leverage a motivating observation of the topology and routing mechanisms of Chord. In a 2^m-node Chord, when a source node broadcasts a large file such as software patches to all other nodes, each node can appropriately select two of m finger nodes as its children, instead of selecting one or all finger nodes. The two-choice approach guarantees that each node has at most two children and all nodes are traversed. Thus, the basic idea behind our algorithms is that a balanced DBT is implicitly constructed from the novel routing paths of Chord, without explicit parent–child membership maintenance. According to the node identifier distribution of Chord, we propose two broadcast algorithms over DHT. (1) When nodes are uniformly distributed in the identifier space, a token-based broadcast algorithm is proposed, where each node selects the finger nodes as its children by a token value. (2) When nodes are arbitrarily distributed in the identifier space, a partition-based broadcast algorithm [33] is proposed, where each node partitions its identifier space into two subspaces and selects the agent nodes in the subspaces as its children. The partition-based algorithm generalizes the token-based algorithm, and is also suitable for the uniform distribution of node identifiers. We show theoretically experimental results show that both token-based and partition-based algorithms can construct and maintain a scalable and balanced DBT, where the branching factor is at most two and the tree height is $O(\log n)$ in a Chord of n nodes, and needs no extra space for storing children and no additional overhead for explicitly maintaining the parent–child membership.

The rest of the paper is organized as follows. Section 2 introduces the background of Chord and typical DHT-based broadcast algorithms. In Section 3, we describe in detail our token-based and partition-based broadcast algorithms over DHT. Section 4 presents our experimental results. Related work is introduced in Sections 5 and 6 and concludes the paper.

2. Background

2.1. Chord overview

The Chord network is modeled as an undirected graph $G = (V, E)$, where the vertex set V contains n nodes and E is the set of overlay links between nodes. Chord utilizes a SHA-1 hash function to assign m-bit identifiers to both data objects and participating nodes. A node's identifier is produced by hashing the node's IP address, while an object's identifier is produced by hashing the object's key. Both objects and nodes have the same identifier space $[0, 2^m)$, and identifiers are ordered in an identifier circle modulo 2^m. According to the node identifiers, Chord organizes nodes as a ring topology in the circular space. An object's identifier k is assigned to the first node whose identifier is equal to or follows k in the identifier space. This node is called the successor node of the identifier k, denoted by $Succ(k)$. In Chord, $Pred(u)$ refers to the immediate predecessor of node u, while $Succ(u)$ refers to the immediate successor of node u. Besides its immediate predecessor and successor, each node u maintains a set of m finger nodes that are spaced exponentially in the identifier space. The ith finger node $Finger(u, i)$ of node u is the first node that succeeds u by at least 2^i in the identifier space, that is $Finger(u, i) = Succ((u + 2^i) \bmod 2^m)$, where $0 \leq i \leq m - 1$.

Chord adopts a greedy finger routing algorithm [23] to recursively (or iteratively) forward a query message with an object's identifier k to its successor node $Succ(k)$ that contains a pair (k, v), where v is the object's value. When node u wants to lookup an object's identifier k, it forwards a query message with the identifier k to its finger node $Finger(u, j)$, which is closest to the successor node